AI Acceleration with Deep Learning Compiler

By TBD: WENSHENG ZHENG SONGGEN YIN RUIXI LIU SUK BAN

With Prof. YongJun Park

Hardware

```
TBD@CASSLAB-Server15:~$ gcc check.c -o check
TBD@CASSLAB-Server15:~$ ./check
Component   | Model
----------|---------------------------
CPU         | Intel(R) Xeon(R) Silver 4214R CPU @ 2.40GHz
GPU         | NVIDIA GeForce RTX 3090
TBD@CASSLAB-Server15:~$
```

```
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 535.146.02             Driver Version: 535.146.02   CUDA Version: 12.2     |
|-------------------------------+----------------------+----------------------+
| GPU  Name                 Persistence-M | Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp   Perf          Pwr:Usage/Cap |         Memory-Usage | GPU-Util  Compute M. |
|                                         |                      |               MIG M. |
|===============================+======================+======================|
|   0  NVIDIA GeForce RTX 3090        On  | 00000000:3B:00.0 Off |                  N/A |
|  0%   30C    P8              30W / 350W |      1MiB / 24576MiB |      0%      Default |
|                                         |                      |                  N/A |
+-------------------------------+----------------------+----------------------+
|   1  NVIDIA GeForce RTX 3090        On  | 00000000:AF:00.0 Off |                  N/A |
|  0%   33C    P8              26W / 350W |      1MiB / 24576MiB |      0%      Default |
|                                         |                      |                  N/A |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
|        ID   ID                                                   Usage      |
|=============================================================================|
|  No running processes found                                                 |
+-----------------------------------------------------------------------------+
```

Hardware Overview: NVIDIA RTX 3090 GPU vs Intel Xeon Silver 4214R CPU

NVIDIA RTX 3090 GPU:

- Built for extensive parallel computation with over 10,000 CUDA cores.
- Highly effective for deep learning workloads, such as matrix multiplications and convolutions.
- Tensor cores and high memory bandwidth accelerate training and inference, optimizing performance for computationally intensive neural networks.

Intel Xeon Silver 4214R CPU:

- Features 12 cores with 24 threads, clocked at 2.40 GHz.
- Designed for balanced performance with a focus on high single-thread performance and complex logic tasks.
- Supports SIMD and Hyper-Threading, but is less efficient for highly parallel tasks compared to GPUs.

Raw Performance comparison

CPU:

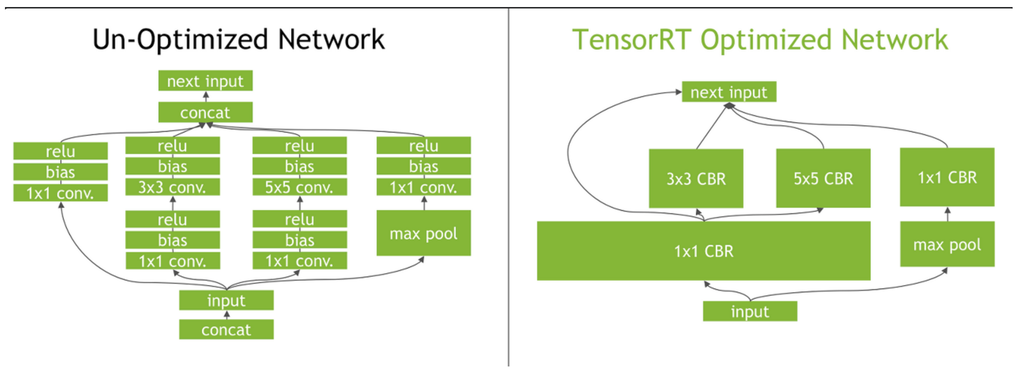
- Fewer cores (usually 4-16).
- Optimized for complex logic and branching tasks.
- Better for single-threaded performance.

GPU:

- Contains many small cores.
- Designed for high parallelism.
- Ideal for deep learning tasks involving independent operations.

```
TBD@CASSLAB-Server15:~/onnx114514$ ls
bench.py   dog.jpeg                    ir.py      resgpu.py          resn
et50_relay_ir.txt
bench2.py  imagenet-simple-labels.json  rescpu.py  resnet50-v2-7.onnx  test
.py
TBD@CASSLAB-Server15:~/onnx114514$ python3 bench.py
+--------+-----------------------------+-------------------------------+
| Device |  Average Time per Inference  |  Total Time for 100 Inferences  |
+--------+-----------------------------+-------------------------------+
|  CPU   |          23.14 ms           |             2.31 s             |
| ------ | --------------------------- | ------------------------------ |
|  GPU   |          17.42 ms           |             1.74 s             |
+--------+-----------------------------+-------------------------------+
```

Kernel Fusion



Un-Optimized Network	TensorRT Optimized Network
Redundant operations like ReLU, bias, convolution.	Combines layers into optimized blocks.
High computational overhead and memory access.	Fused operations reduce overhead.
Complex computational graph, harder to optimize.	Simplified graph for efficient scheduling.
Inefficient execution, increased latency.	Better hardware utilization, reduced latency.

Kernels are executable code segments and kernel fusion is a technique for combing the segments in a coherent manner to improve execution time. Fusing image processing kernels to be executed on GPUs for improving execution time and total throughput (amount of data processed in unit time).

Project Outcomes

Model performance after AI-acceleration optimization under different frameworks:

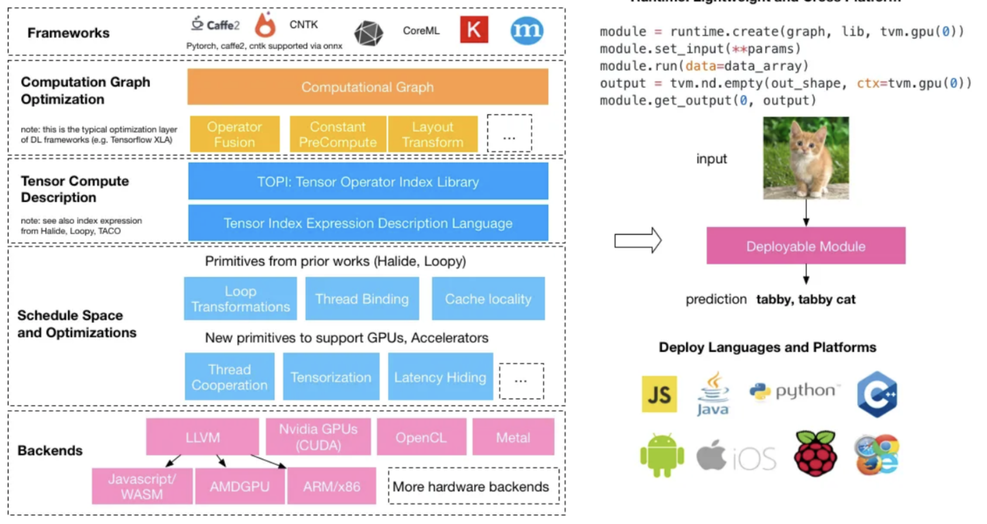
```
TBD@CASSLAB-Server15:~/onnx114514$ python3 bench2.py
+----------------+----------------+-----------------------------+
|     Model      |  Average Time  | Total Time (100 Inferences) |
+----------------+----------------+-----------------------------+
|    resnet50    |    6.67 ms     |           0.67 s            |
| -------------- | -------------- |  -------------------------  |
|      tvm       |    6.67 ms     |           0.67 s            |
| -------------- | -------------- |  -------------------------  |
|  tvm op_fused  |    4.90 ms     |           0.49 s            |
| -------------- | -------------- |  -------------------------  |
|      tvm2      |    4.65 ms     |           0.47 s            |
| -------------- | -------------- |  -------------------------  |
|      tvm3      |    2.03 ms     |           0.20 s            |
| -------------- | -------------- |  -------------------------  |
|   onnx (gpu)   |    10.6 ms     |           1.06 s            |
| -------------- | -------------- |  -------------------------  |
|      cpu       |    24.78 ms    |           2.48 s            |
+----------------+----------------+-----------------------------+
TBD@CASSLAB-Server15:~/onnx114514$
```

Key Optimizations for Faster Execution

- **OpFusion**: Combines operations to reduce kernel launches and memory overhead.
- **FoldConstant**: Computes constants at compile time, reducing runtime calculations.
- **FoldScaleAxis**: Merges scaling operations, simplifying the computational graph.
- **AlterOpLayout**: Changes memory layout to improve cache efficiency and hardware performance.
- **CanonicalizeOps**: Simplifies operations to make them faster and easier to optimize.
- **CanonicalizeCast**: Eliminates unnecessary type conversions for reduced processing overhead.
- **EliminateCommonSubexpr**: Removes duplicate calculations to avoid redundancy.

A performance table of the model on the GPU across different frameworks. By using TVM to optimize the model, we observe a 5x speedup in performance with the applied optimizations. Specifically, we utilized a combination of OpFusion, FoldConstant, FoldScaleAxis, AlterOpLayout, CanonicalizeOps, CanonicalizeCast, and EliminateCommonSubexpr to enhance the model's efficiency.

TVM diagram



Integration with Frameworks	Works with PyTorch, TensorFlow to optimize models.
Computation Graph Optimization	Optimizes computation graphs for better efficiency.
Tensor Expression Generation	Generates optimized tensor expressions for deployment.
Advanced Scheduling	Uses scheduling to achieve hardware acceleration.
Multiple Backend Support	Supports NVIDIA GPUs, ARM CPUs, and web platforms.
Deployable Modules	Produces lightweight modules for Python, JavaScript, etc.

References

[1] Tianqi Chen, Thierry Moreau, Ziheng Jiang, Eddie Zheng, Haichen Yan, Leyuan Shen, Milo Cowan, Lianmin Wang, Yuwei Hu, Luis Ceze, and others.
TVM: An Automated End-to-End Optimizing Compiler for Deep Learning.
arXiv preprint arXiv:1802.04799, 2018.

[2] Zhi Wang, Wei Wei, Ling Wang, Yiqun Wang, Ji Zhou, and Huazhong Yang.
The Deep Learning Compiler: A Comprehensive Survey.
arXiv preprint arXiv:2002.03794, 2020.

[3] NVIDIA.
NVIDIA Deep Learning Accelerator (NVDLA).
2021.
Available at: `https://nvdla.org/`.

[4] Sanket Tavarageri, Gagandeep Goyal, Sasikanth Avancha, Bharat Kaul, and Ramakrishna Upadrasta.
AI Powered Compiler Techniques for DL Code Optimization.
arXiv preprint arXiv:2104.05573, 2021.

[5] Sean Kinzer, Soroush Ghodrati, Rohan Mahapatra, Byung Hoon Ahn, Edwin Mascarenhas, Xiaolong Li, Janarbek Matai, Liang Zhang, and Hadi Esmaeilzadeh.
Restoring the Broken Covenant Between Compilers and Deep Learning Accelerators.
arXiv preprint arXiv:2310.17912, 2023.

[6] NVIDIA.
NVDLA Deep Learning Inference Compiler is Now Open Source.
2019.
Available at: `https://developer.nvidia.com/blog/nvdla/`.